LEA 是微机 8086/8088 系列的一条指令，取自英语 Load effect address——取有效地址，也就是取偏移地址。在微机 8086/8088 中有 20 位物理地址，由 16 位段基址向左偏移 4 位再与偏移地址之和得到。

取偏移地址指令

指令格式如下：

LEA reg16,mem

LEA 指令将存储器操作数 mem 的 4 位 16 进制偏移地址送到指定的寄存器。这里，源操作数必须是存储器操作数，目标操作数必须是 16 位通用寄存器。因该寄存器常用来作为地址指针，故在此最好选用四个间址寄存器 BX,BP,SI,DI 之一。

LEA 取有效地址指令 （Load Effective Address ）

指令格式：LEA 目的，源

指令功能：取源操作数地址的偏移量，并把它传送到目的操作数所在的单元。

LEA 指令要求原操作数必须是 存储单元 ，而且目的操作数必须是一个除段寄存器之外的 16 位或 32 位寄存器。当目的操作数是 16 位通用寄存器时，那么只装入有效地址的低 16 位。使用时要注意它与 MOV 指令的区别，MOV 指令传送的一般是源操作数中的内容而不是地址。

例 1 假设：SI=1000H , DS=5000H, (51000H)=1234H

执行指令 LEA BX , [SI]后，BX=1000H

执行指令 MOV BX , [SI]后，BX=1234H

有时，LEA 指令也可用取偏移地址的 MOV 指令替代。

例 2 下面两条指令就是等价的，他们都取 TABLE 的偏移地址，然后送到 BX 中，即

LEA BX,TABLE

MOV BX,OFFSET TABLE

但有些时候，必须使用 LEA 指令来完成某些功能，不能用 MOV 指令来实现，必须使用下面指令：

LEA BX, 6[DI]

解释：某 数组 含 20 个元素，每个元素占一个字节，序号为 0~19。设 DI 指向数组开头处，如果把序号为 6 的元素的偏移地址送到 BX 中

八进制	十六进制	十进制	字符		八进制	十六进制	十进制	**字符**
0	0	0	nul		100	40	64	@
1	1	1	soh		101	41	65	A
2	2	2	stx		102	42	66	B
3	3	3	etx		103	43	67	C
4	4	4	eot		104	44	68	D
5	5	5	enq		105	45	69	E

6	6	6	ack		106	46	70	F
7	7	7	bel		107	47	71	G
10	8	8	bs		110	48	72	H
11	9	9	ht		111	49	73	I
12	0a	10	nl		112	4a	74	J
13	0b	11	vt		113	4b	75	K
14	0c	12	ff		114	4c	76	L
15	0d	13	cr		115	4d	77	M
16	0e	14	so		116	4e	78	N
17	0f	15	si		117	4f	79	O
20	10	16	dle		120	50	80	P
21	11	17	dc1		121	51	81	Q
22	12	18	dc2		122	52	82	R
23	13	19	dc3		123	53	83	S
24	14	20	dc4		124	54	84	T
25	15	21	nak		125	55	85	U
26	16	22	syn		126	56	86	V
27	17	23	etb		127	57	87	W
30	18	24	can		130	58	88	X
31	19	25	em		131	59	89	Y
32	1a	26	sub		132	5a	90	Z
33	1b	27	esc		133	5b	91	[
34	1c	28	fs		134	5c	92	\
35	1d	29	gs		135	5d	93	]
36	1e	30	re		136	5e	94	^
37	1f	31	us		137	5f	95	_
40	20	32	sp		140	60	96	'
41	21	33	!		141	61	97	a
42	22	34	"		142	62	98	b
43	23	35	#		143	63	99	c
44	24	36	$		144	64	100	d
45	25	37	%		145	65	101	e

46	26	38	&		146	66	102	f
47	27	39	`		147	67	103	g
50	28	40	(		150	68	104	h
51	29	41	)		151	69	105	i
52	2a	42	*		152	6a	106	j
53	2b	43	+		153	6b	107	k
54	2c	44	,		154	6c	108	l
55	2d	45	-		155	6d	109	m
56	2e	46	.		156	6e	110	n
57	2f	47	/		157	6f	111	o
60	30	48	0		160	70	112	p
61	31	49	1		161	71	113	q
62	32	50	2		162	72	114	r
63	33	51	3		163	73	115	s
64	34	52	4		164	74	116	t
65	35	53	5		165	75	117	u
66	36	54	6		166	76	118	v
67	37	55	7		167	77	119	w
70	38	56	8		170	78	120	x
71	39	57	9		171	79	121	y
72	3a	58	:		172	7a	122	z
73	3b	59	;		173	7b	123	{
74	3c	60	<		174	7c	124	|
75	3d	61	=		175	7d	125	}
76	3e	62	>		176	7e	126	~
77	3f	63	?		177	7f	127	del

SHL、SHR、SAL、SAR: 移位指令

;SHL(Shift Left): 逻辑左移
;SHR(Shift Right): 逻辑右移
;SAL(Shift Arithmetic Left): 算术左移
;SAR(Shift Arithmetic Right): 算术右移

;其中的 SHL 和 SAL 相同，但 SHR 和 SAR 不同.

;SHL、SAL: 每位左移，低位补 0，高位进 CF
;SHR : 每位右移，低位进 CF，高位补 0
;SAR : 每位右移，低位进 CF，高位不变

;它们的结果影响 OF、SF、ZF、PF、CF

ROL、ROR、RCL、RCR: 循环移位指令

;ROL(Rotate Left): 循环左移
;ROR(Rotate Right): 循环右移
;RCL(Rotate through Carry Left): 带进位循环左移
;RCR(Rotate through Carry Right): 带进位循环右移

;ROL: 循环左移，高位到低位并送 CF
;ROR: 循环右移，低位到高位并送 CF
;RCL: 循环左移，进位值(原 CF)到低位，高位进 CF
;RCR: 循环右移，进位值(原 CF)到高位，低位进 CF

;它们的结果影响 OF、CF